04

Entidades JPA

Transcripción

[00:00] Ya tenemos conectada nuestra base de datos. Tenemos todo para comenzar a trabajar. ¿Qué es lo que sigue? Tenemos que definir el modelo que vamos a utilizar para la persistencia, el modelo de tablas, la relación entre las entidades. Para esto, vamos a comenzar creando en el paquete médico nuestra clase que va a mapear la entidad médico.

[00:23] Esto no es un curso de JPA entonces vamos a hacerlo lo más directo posible. Para eso vamos a crear una nueva clase llamada médico, voy a agregar aquí y médico va a tener sus atributos. Por ejemplo, si no recuerdo los atributos serían básicamente los mismos que están aquí en el DatoRegistroMédico.

[00:47] La diferencia es DatosRegistroMedico es nuestro DTO, Data Transfer Object, que es usado para recibir lo que está viniendo del API y nuestra entidad médico que estamos creando que vamos llenar los atributos ahora, es la que vamos a usar para hacer la persistencia de datos con nuestro modelo de base de datos.

[01:09] Son atributos similares, pero utilidades muy diferentes, no lo olviden. Incluso ni siquiera necesitan ser iguales, hay casos en los que tú no necesitas exponer todos los atributos que tienes en tu entidad y el DTO te sirve para justamente solo exponer lo que tú quieres exponer.

[01:28] Sin más preámbulos, vamos a comenzar a crear. Primero un id porque es una entidad. Un id, vamos a dar un private String nombre, private String

email, también necesitamos un private String documento; me equivoqué, hay un typo. Listo. También tenemos la especialidad, ella es un enum, private Especialidad especialidad. Muy bien.

[02:03] Y finalmente tenemos el private Dirección, nuevamente el tipo, Dirección. No existe aún, lo vamos a crear, tamaño dirección. Como dirección no existe lo que voy a hacer es darle clic en la flechita roja y crear class dirección, obviamente en el paquete dirección.

[02:25] Creó aquí. No lo agregó aún. Y aquí path simplemente con los atributos private String calle; private Integer número; private String complemento; private String ciudad y no recuerdo si me estoy olvidando de algo, entonces voy a entrar a DatosDirección es calle, distrito. Falta distrito. Entonces regreso aquí a mi entidad, copio esto y aquí le voy a poner distrito.

[03:00] Y ya tengo mapeados los atributos que llegan a mi DTO que sería ese ahí, pero ahora mi clase de persistencia. Y no puede ser una clase de persistencia, si no tienes las anotaciones de persistencia que sería en este caso table de persistence. Aquí le voy a poner el nombre de tabla que va a ser "médicos" y finalmente entity, que el nombre de mi entidad va a ser "Médico". Aquí faltó el atributo, listo.

[03:42] ¿Qué más está faltando? Falta indicarle el id. Nuevamente no vamos a profundizar en esto porque no es un curso de JPA, solamente voy a completar eso. El strategy. En GenerationType va a ser identity, para el id, aquí va ser enumerated, y el EnumType va a ser un string y esto va a ser embedded. ¿Por qué?

[04:22] Porque lo que estamos haciendo en nuestro formulario tenemos todos los datos juntos, tenemos los datos del médico y la dirección, lo mismo lo vamos a plasmar en la base de datos, pero como ya vimos que Dirección lo va a usar tanto el paciente como el médico, no tiene sentido duplicar los mismos atributos en dos entidades, por eso lo estamos dividiendo.

[04:46] Y lo que vamos a hacer es ahora anotar dirección con embeddable. Muy bien. No es una tabla. Es solamente una clase que es embeddable en la otra para que la incluya nada más. Yy con esto ya tendríamos listo lo que es nuestro modelo, nuestra definición de JPA para mapear nuestras entidades.

[05:13] Pero como toda clase, necesitamos crear constructor, getter y yo no quiero tener que escribir a mano ese código repetitivo, no tiene caso, no tiene sentido para mí y es aquí donde entra Lombok en acción, ¿por qué? [05:32] Porque como les dije antes, Lombok lo que hace es generar automáticamente el código que no tiene sentido escribirlo por ti mismo, por ejemplo, si tú quieres generar getters, le vas a poner la opción getter de Lombok y automáticamente una vez que compile, Lombok va a generar automáticamente los getters para todos estos atributos.

[05:55] Es más, necesito un constructor sin atributos, entonces le va a poner NoArgsConstructor para un constructor default y un constructor con todos los atributos pongo AllArgsConstructor. Y finalmente aquí lo que necesito, déjame buscar un poco aquí, es implementar mi EqualsAndHashCode.

[06:20] EqualsAndHashCode lo que va a hacer es usar el parámetro id para las comparaciones entre médicos, por ejemplo, aquí add of, listo. Entonces dependiendo del parámetro id, si los id son iguales va a inferir que son iguales. Como si yo sobrescribiera el método hashcode y el método equals en mi clase, pero Lombok lo va hacer por mí.

[06:48] De igual forma le vamos a hacer para dirección. EqualsAndHashCode por qué dirección no tiene id. Lo copio y no hay ningún problema Así es cómo ya tengo finalmente mis entidades mapeadas listas para implementar mi persistencia en la base de datos. ¿Pero qué es lo que está faltando aquí?

[07:13] Vamos al MedicoController. Hasta ahora, una vez que recibimos los datos del registro médico, solo los estamos imprimiendo en la terminal. ¿Qué es lo que debemos hacer ahora? Debemos crear una capa extra en la que vamos

a tomar los datos que llegan, transformarlos, vamos a transformarlos a nuestras entidades y estas entidades van a persistir en la base datos.

[07:39] Antiguamente se usaba lo que es el patrón DAO, Data Access Object en el cual tú tenías que crear tu entity manager, gestionar la sesión. Y si usabas Hybernate crear tu prepared statement, todo lo que eso conlleva, un código extenso que la verdad que no lo recuerdo de memoria, es muy extenso para recordarlo, necesitaría ver documentación.

[08:04] Pero ya ha pasado muchos años de esos tiempos, ahora hay formas mucho mejores de hacerlo, obviamente Spring nos hace la vida mucho más fácil. Es el tema del siguiente video.